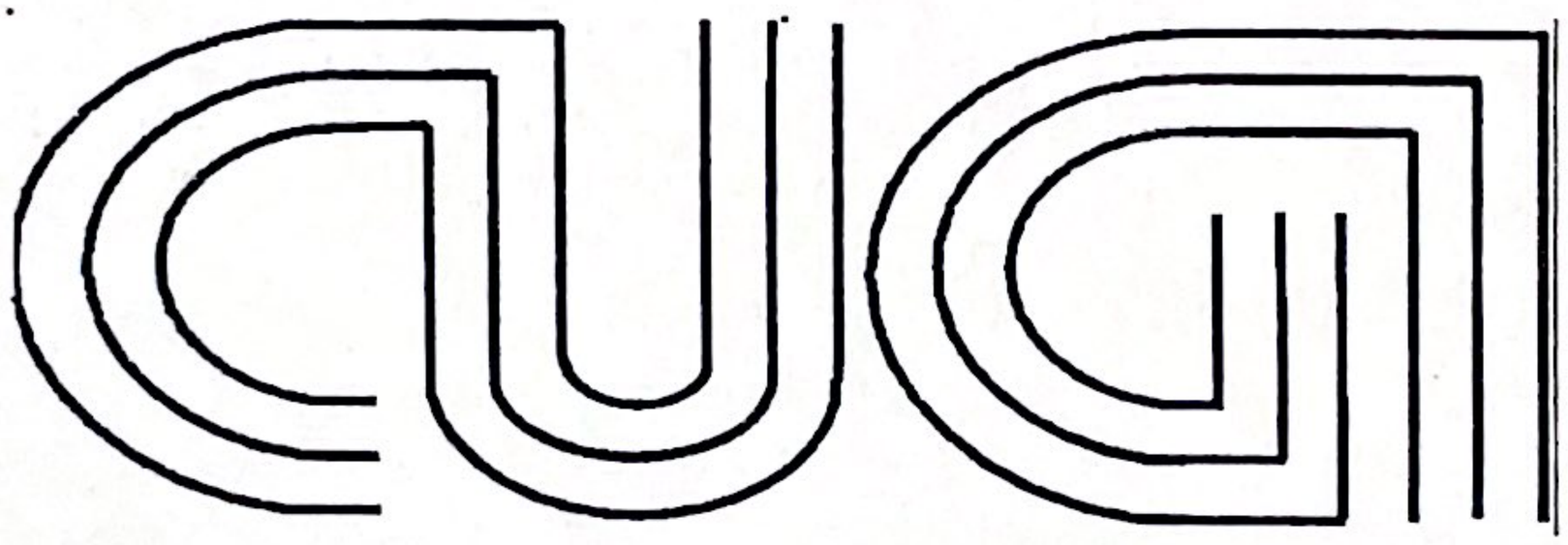


Newsletter
October 1995

In this issue...

HONOURARY CUGI OFFICIALS

Chairman	Karl Jeacle kj@broadcom.ie
Treasurer	Colin Dalton csdalton@tcd.ie
Secretary	Kevin Phair kev@scorpio.ie
Communications Officer	Matt Brookes mtb@broadcom.ie
Newsletter Editor	Gavin McConnon gav@alien.ie
Librarian	David Fitzgerald
Technical Consultant	Eddy Carroll ecarroll@iol.ie

CUGI meetings are held on the third Sunday of each month from 3 pm to 6 pm in Room 54 at St Andrew's College, Booterstown. Correspondence should be addressed to:

CUGI
c/o St Andrew's College,
Booterstown,
Co Dublin

If you have a modem, call CUGI BBS at (01) 837 0204. You can reach CUGI by Internet mail at cugi@ieunet.ie, or individual committee members at the network addresses listed above.

Newsletter

Contents

Volume 7 Number 4
October 1995

The deadline for articles for the next Newsletter is January 5, 1996

Editorial

by Gavin McConnon

WELCOME to the October Newsletter. Yes we're a bit late, but with the AGM last month and some problems getting articles over the Internet, things were a little chaotic. Also, being new at this didn't help the timing —thanks for all the help Colin. Anyway, we have to keep to the late newsletter tradition don't we :-)

Well another CUGI year has passed and another committee assigned their roles. With Eddy stepping down after seven years of excellent work as Editor, I was nominated to fill his shoes. I didn't realise how much time and work was involved in producing the newsletter—so get your articles in early!

This is the first newsletter to feature our new style. We are using desktop publishing software rather than TEX, this gives us more flexibility with graphics and page layout. The biggest change is probably two columns of text, and you probably will not notice the rest!

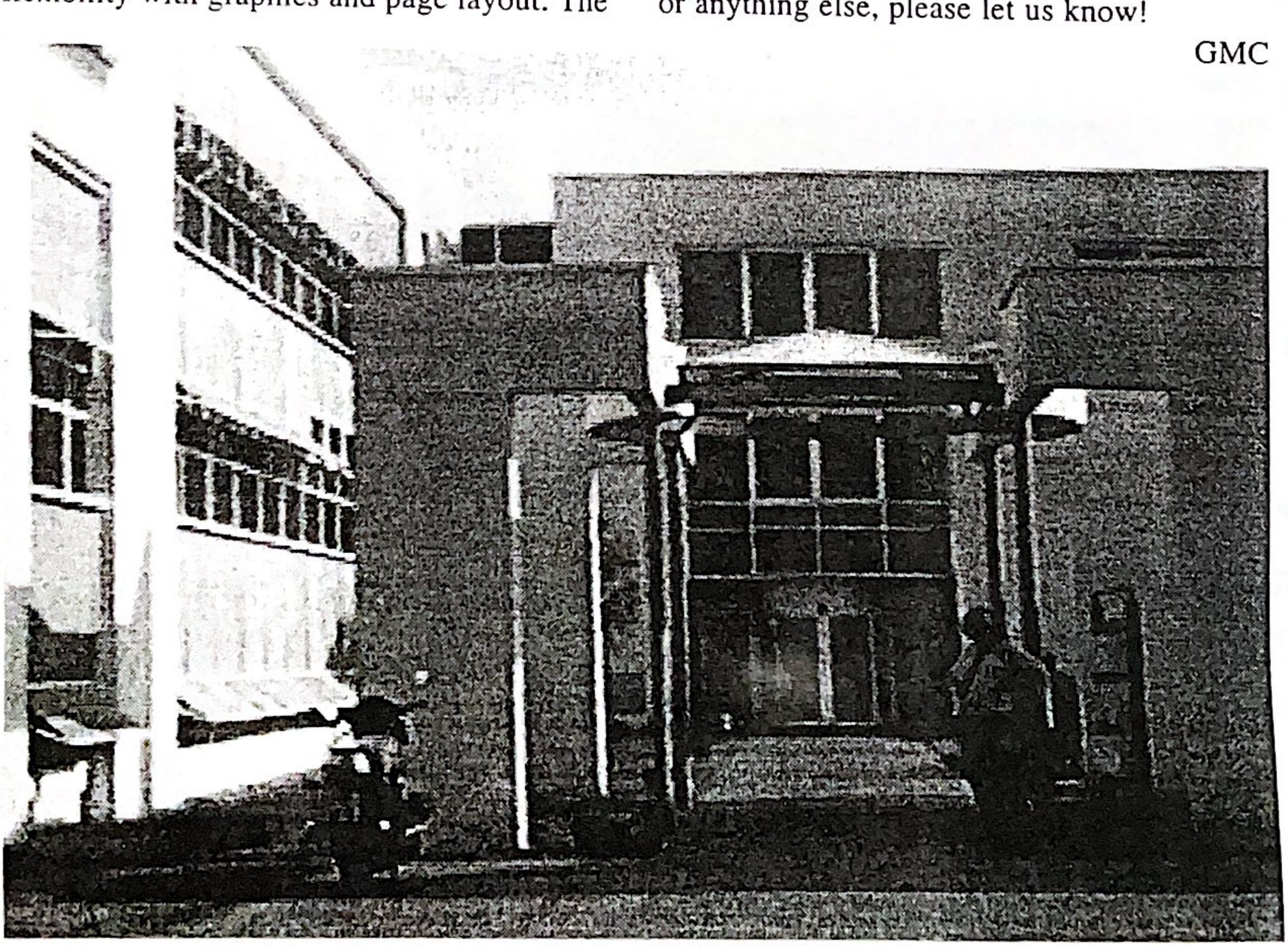
At last some Amiga Technologies news to report, along with some trade show news, check out Colin's in-depth article. The World Wide Web is dressing up with Java, as Karl keeps us up to date with the fast-changing Internet world.

Be Inc. have released their BeBox, which looks set to be an affordable hackers' (in the original sense of the word) machine. Check out Eddy's article on page 17.

If you have found a new utility you like, or bought some new hardware, please write us an article. You can give articles to anyone on the committee.

Finally, if you have any ideas for the newsletter, whether it be an article style change or anything else, please let us know!

GMC



Dr. Peter Kittel outside Amiga Technologies' HQ in Benshein Germany

eXciting Windows?

by Colin Dalton

THE X Window System is a retargetable graphics system developed at MIT in the 1980s. It allows you, among other things, to run a program on one machine, and have its output displayed on another. X is split into client and server sides. The server is responsible for managing requests from clients to display on the host's screen, and the clients are the programs that you run.

There is also a special kind of client called the Window Manager. This software is responsible for managing the "Look and Feel" of the desktop; how the window borders, close gadgets and other screen furniture looks and behaves.

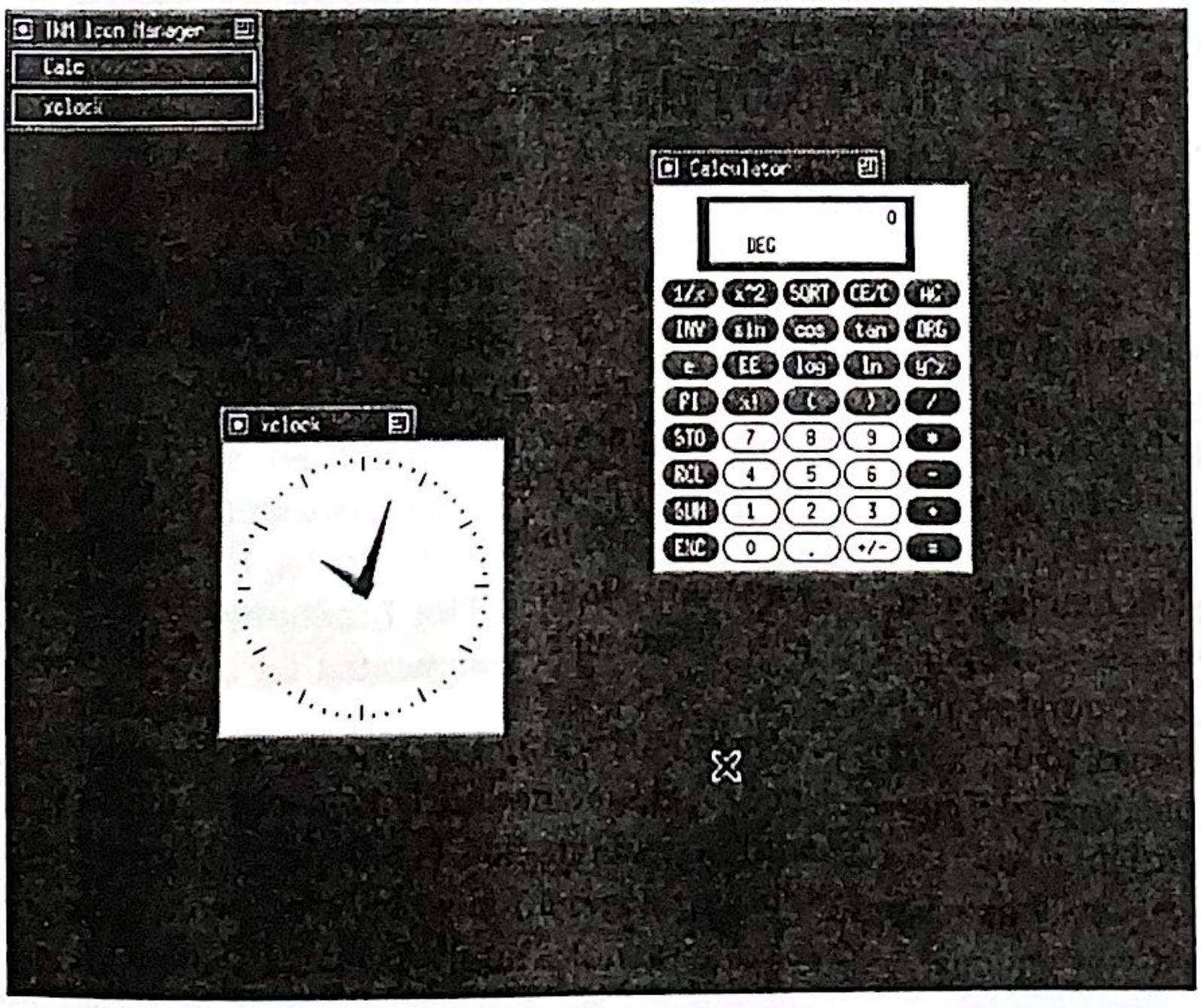
The X Window System is device and platform independent, as long as you have an X server compiled for your machine, you can display X programs. However, most X software runs on Unix machines, so this means you can operate and view heavyweight Unix software on a machine with minimal RAM, CPU and disk resources.

The Amiga has (or had) three X servers packages. The first two, DaggeX and the X11 server from GfxBase are, to my knowledge, no longer in development, however a third, AmiWin, recently joined the fray. AmiWin is a fully-featured X11R6 server for the Amiga, sold using the shareware system. In the demonstration version, you are limited to screens of 800x600 or less, and you cannot have more than four clients connected at the same time.

Now with AmiWin, it is possible for Amiga users to view Netscape on their own machines. Assuming you have a fast enough connection to a Unix machine, you can run Netscape on it, and redirect the display to your Amiga. The same applies for DOOM, the inane game that seems to have half the PC owners on the planet hooked. Conversely, it is also possible to view X programs being run on an Amiga on another X terminal, be it a dumb terminal or an Alpha-based workstation. However, unless you port some Amiga application to use X, this feature is of limited use, as AmiWin comes with only a few demonstration X clients, such as xlogo, xclock, and xcalc, and the basic twm window manager.

In use, AmiWin has proved itself reliable and compatible, which is more than can be said of certain PC X servers, costing a lot more. On the subject of cost, the full registered package costs $50, and includes more display drivers, fonts and clients.

The demo version of AmiWin is available from any Aminet site, in the gfx/x11 directory.



X11, with the twm window manager running on an Amiga. The calculator program is actually running on a Unix machine.

Java: Programming for the Internet

by Karl Jeacle

IF this is the first time you've come across Java, I can guarantee you, it won't be the last. Over the next twelve months, just about every Internet related-publication you read will have something to say about Java.

Sun Microsystem's Java programming language and HotJava™ WWW browser are taking the Internet by storm. Since its release in April 1995, Java has started breathing new life into the World Wide Web, which from a user perspective, has remained largely static in technology terms over the last few years.

So what is it?

Java is a programming language. HotJava is a Web browser. HotJava is just like any other Web browser except it can display HTML pages which contain not just text and images, but small applications, or "applets". This "executable content" runs on the Web page. HotJava first downloads a page's HTML file, formats it, inserts any images, and then starts running any applets present on the page. These applets can be as simple as a small animation or audio player "live" on a page, or as complex as any word processor or spreadsheet. Java applets know no bounds.

HotJava is a called a dynamic browser; for a number of reasons. In addition to the interactive content described above, its late binding system offers dynamic types and dynamic protocols.

While most browsers are monolithic programs, with support for HTML, HTTP, and GIF all compiled in, HotJava is dynamic in nature. If a conventional browser has no support for something like JPEG, a new release of the browser is required. With HotJava, if new formats appear, the browser downloads the JPEG file from a Web server, realises it doesn't understand the type, and queries the Web server for code to handle the JPEG file.

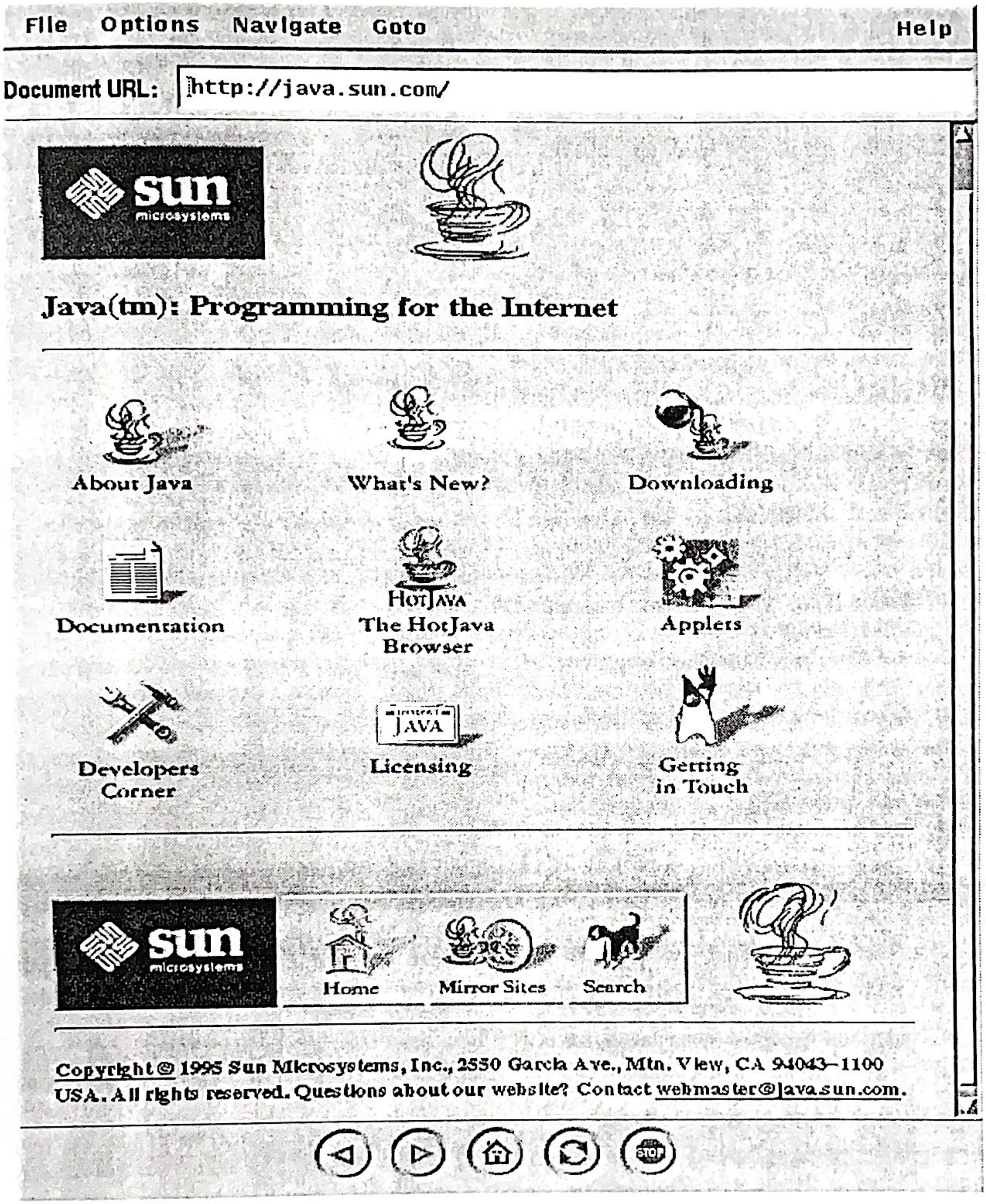
The same scenario applies when dealing with new protocols. HTTP is, at present, the most common protocol used on the Web. New protocols are being developed which are more secure, which can be used for Internet commerce. The drawback with these new protocols is that both client browser and Web server software must be aware of them. So if Bank A decides to use Company X's secure Web server, and Bank B decides to use Company Y's secure Web server, both speaking their own proprietary secure versions of HTTP, customers will have to use two different browsers. With HotJava, a new protocol can be added dynamically in the same way as with dynamic types.

History

The origins of the Java project stemmed from a desire by Sun to build advanced software for a wide variety of networked devices, embedded systems, and even consumer electronics devices. The goal was to build a small, distributed, real-time operating environment. C++ was the original language of choice, but as time went by, the difficulties encountered using C++ for the project grew to the point where it made more sense to develop a whole new language, than continue with C++. Hence Java.

If you're wondering how the name Java came about, well, join the club. There have been a lot of stories on the Net about where the name came from. About the most reliable I've come across so far is from Tim Lindholm of Sun Microsystems, who commented on a Usenet news group:

"No, Java is not an acronym. It was come up with in a room full of frustrated people trying to find something that sounded OK and hadn't been trademarked already. The original name of the language (Oak) was owned by someone else and could not be used."



The HotJava browser in action

In Sun's White Paper on Java, they use a set of buzzwords to describe Java. However, they then go on to explain each of these one by one in order to present a picture of the problems they were trying to solve when designing Java. I'll try to summarize this, but first, the buzzwords:

> *Java: A simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic language.*

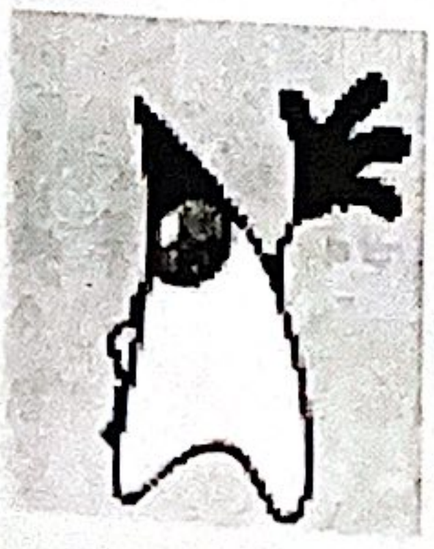
Programming languages are getting larger, more feature-rich, and hence more complex and difficult to use. Java is simple, it's small. Start with C++, remove a bunch of rarely used complicated features, add things like automatic garbage collection (no more `malloc()` and `free()`), and you've got simple and OO. You've also got small. A self-contained Java microkernel can be built in less than 220 kilobytes.

Distribution: Java provides libraries with TCP code for handling protocols like HTTP and FTP. Java is intended for writing programs that will be reliable. A lot of emphasis goes on early checking for potential problems and avoiding situations which are error prone. One example of this is Java's pointer model. No more pointer arithmetic, Java has true arrays. Programmers can worry less about corrupting memory. This helps make Java robust.

Since Java is intended for use in a networked/distributed environment, a lot of emphasis has been put on security. The pointer model described above does not allow applications to gain access to object data structures to which they have no access. Combined with public-key encryption techniques, this helps the robustness and secureness of Java.

Networked environments contain lots of different types of equipment. To build a system which can be used to develop code which runs on all of these, it's got to be both architecture neutral and portable. Java applets will run on anything which has a port of the Java engine. Once you've got this engine, it runs your applets by interpreting the bytecodes created by the Java compiler. Java is interpreted.

Despite the interpreted nature of Java, Sun have built the bytecode system so as to gain as high a performance run-time system as possible. To increase interactive responsiveness and real-time behaviour, support for threads has been built into Java. In a world where many different events are happening simultaneously, no longer does the programmer have to link in third party threads packages to handle multiple simultaneous tasks. Java inherently supports multithreading.



Finally, Java is a dynamic language. It was designed to adapt to an evolving environment. By using a concept known as interfaces, and by binding software component objects together as late as possible, Java avoids a number of software reusability problems evident with languages such as C++, and makes the use of the OO paradigm much more of a reality.

Using Java

Programming Java is a lot like C++. If you can program in C++, you should feel comfortable programming Java within a few days. If you only know C, things are a little harder, or at least, unfamiliar. Once you get used to thinking in terms of objects, and working with object-oriented techniques, you should be fine - you don't need to learn C++ first.

As an example, take a look at the classic "Hello World" program written in Java:

```
import browser.Applet;
import awt.Graphics;
class HelloWorld extends Applet {
public void init() {
resize(150, 25);
      }
public void paint(Graphics g) {
g.drawString("Hello world!", 50,
25);
      }
}
```

This code creates a program, or applet in Java terms, which is 150 by 25 pixels in size, and paints a message at co-ordinates (50,25). Every applet has an init() routine which is called when the applet is created. The paint() routine is called whenever a redraw or refresh is required. In this applet, it always prints the same thing, however, if there was a slider in the applet which the user could vary some value as user input, the paint() routine could be used to dynamically calculate the applet display to reflect the position of the slider. I've over-simplified things here a little, but you should get the general idea.

To get this applet up and running on your

Web page, you would save it from your editor as "hello.java", and run the code through the Java compiler by typing something like "javac hello.java". The compiler then outputs a file called "hello.class" which is the binary bytecode file. You're all set. In the same way as there's a HTML tag for including pictures (<IMG SRC="">), there's a tag to place applets in your code. This is either APP or APPLET, depending on which release of Java you're using.

Platforms supported

The initial version of HotJava ran only on Sun SPARCstations running Solaris 2.x. Since then, Sun have released HotJava for Windows NT and Windows 95. This version of the software is called the 1.0Alpha3 release. Since the goal of the Alpha release was to demonstrate the concept of executable content, and to allow developers become familiar with Java, the current version of HotJava is also the last. Work on Java itself has already moved on to a 1.0Pre-Beta release. New third-party browsers will all use this new standard.

Netscape Communications Corporation have announced that they have licensed the Java technology from Sun and their WWW browser Netscape Navigator 2.0 will include support for Java applets. In principle, this means Java could be available for Windows 3.1, Windows 95, Windows NT, Macintosh System 7.5, and a number of Unix variants. At the time of writing, Java support was only present in Sun Solaris and Silicon Graphics IRIX versions.

I know what you're thinking: this all sounds great, but I didn't see the Amiga mentioned anywhere up there, so can I get it for my Amiga? Well, unfortunately, the answer is no; at least, not yet. If you want to be able to use Java on your Amiga, there are a number of possible solutions:

Get a Macintosh emulator such as ShapeShifter and wait for Netscape to release the Java-compliant Macintosh version of Netscape.

Run NetBSD/Amiga, and wait for the NetBSD crowd to come up with a HotJava or other browser, and then for the Amiga guys to put it on NetBSD/Amiga. Ditto for Linux.

Wait for the Amiga Port. Yes indeed, the Amiga HotJava Porting Project is underway, it's called P'Jami (I really don't like this name!). A Web page (http://www.lls.se/~matjo/PJAmi/PJAmi.html) and mailing-list have been created. To subscribe to the mailing-list, send a one line message of "`subscribe amiga-hotjava joe@bloggs.com`" to `majordomo@mail.imnet.de`, substituting your email address for that of Joe Bloggs.

Further Information

Hopefully this article will have stimulated your curiosity enough to look for further information on Java. The best place to start is Sun's official Java web site (http://java.sun.com/). From here you can find just about everything you need to know. Two other resources worth mentioning are the recently created Usenet newsgroup comp.lang.java, and Gamelon (http://www.gamelan.com/), a directory and registry of Java applets and resources available on the Web. And finally, if you're looking for a good example of a Java-powered Web page, Sun Microsystem's home page always impresses; without Java, it's a nice colour picture; with Java—it's alive!

Conclusion

When I first encountered the Web in 1993, I was hooked. It was one of the most amazing yet simple ideas I had ever come across. Two years later, and my interest was waning. Sure the Web was a great resource, but surfing had lost something of its magical quality.

My first impression on encountering HotJava, like many others, was "Wow!".

I strongly encourage you to see a demo of a Java-capable browser in action. For despite my enthusiasm in this article, reading about it is no substitute for seeing it. Believe me, you won't be disappointed.

Java and HotJava are trademarks of Sun Microsystems, Inc.

Hope for the future

by Colin Dalton

AFTER a long period of apparent inactivity, things are finally moving again in the Amiga world, and all at once, it seems!

First, there was the announcement at the Video Toaster User show in Los Angeles, that Amiga technologies will be using the PowerPC chip for the next generation Amigas. They will be called (surprise surprise), Power Amiga. More surprising is the fact that Amiga Technologies were approached by Apple, with a copy of QuickTime ported to the Amiga. QuickTime is Apple's multimedia movie format (perhaps what CDXL should have been), and a QuickTime player for the Amiga would play an important part in getting the machine accepted as a multimedia platform. There are reports that Apple and Amiga Technologies are also going to cross-licence parts of each others' operating systems, and they have entered into a strategic alliance with them.

The first of the new A4000Ts (which rolled of the production line on October 30th) were also unveiled, and a see-through A1200, with an Amiga Technologies-produced motherboard inside. Petro Tyschtschenko, in an upbeat speech, declared that Escom had not just bought the Amiga "for Christmas", and had already spent many times the initial $12m investment it paid to acquire Commodore. Amiga Technologies' PRO, Gillies Bourdon, also said that they were investigating, with Motorola, the possibility of making a hybrid Power PC chip, that incorporated part of a 68040 core, in order to retain compatibility with older software. The first Power Amigas, sporting the PPC604 chip, are expected in the first quarter of 1997, with PPC accelerator boards becoming available for existing Amigas by the end of 1996. Start saving!

Amiga 1200 owners are also being catered, with news that a PCMCIA fax/modem card is being designed. Apparently, 20,000 new 1200s have been sold to end-users, with a backlog of 60,000 still to be filled. The high price of the 4000T was justified by the fact that it was a new product, being built from the start (Commodore only ever made about 50 units, all hand-built).

Apparently Amiga Technologies aren't the only people making noises about PPC-based Amigas. Just before the VTU exhibition, rumours started circulating about Phase 5 and a PowerPC accelerator board for A4000s. These were confirmed, and it seems work is already underway, with a pre-production board shown at the Amiga show in Cologne. Phase 5 plan on porting Exec (the heart of the Amiga's OS) to native PPC code, and then writing a 680x0 emulator to support the rest of the OS and applications. They also plan on writing a PPC-

Dr Peter Kittel, Amiga Technologies

native version of CyberGraphX, Phase 5's Retargatable Graphics System.

Those wishing for a portable Amiga may have their dreams come true with the PAWS—Portable Amiga WorkStation. They will be sold as kits for A600, A1200 and A4000s. with all but the latter having a battery option. Since the A4000 motherboard draws some 70 watts, it would drain a normal laptop battery in a matter of minutes. However, these kits aren't going to be cheap, with a list price in the order of $3,500, and availability is dependent on FCC approval and finding distributors.

Another show to take place recently was the World of Amiga in Cologne, Germany. Not surprisingly, Amiga Technologies had a large stand there, housing many third party developers as well as themselves. One of the most encouraging reports to come out of this show was the release of a new C/C++ compiler, called StormC. This features intelligent Drag-and-Drop, clickable error messages, easy project management, an editor with syntax highlighting, a debugger which interacts with the editor, and even resource tracking. StormC certainly sounds as if it can easily bridge the gap left by the departure of SAS, and give Amiga developers a badly needed professional development system.

All in all, the past few weeks have been very positive for the Amiga, and finally we have a future to look forward to.

Diary Dates

December 17	Last CUGI meeting of the year, bring your Santa hat!
January 5	Deadline for submission of articles for the next newsletter
January 21	First meeting of the new year!
February 18	First meeting of February - bring your Valentine!

Classified Ads

For sale

Emplant Emulation System. Run Mac and PC software on your Amiga. Comes with SCSI chip, and Macintosh software.
£190 ono. Call Gavin @ 088-623567, e-mail gav@alien.ie.

Two Eurosonic FM CB radios. With car aerial, external power supplies and car power adaptor. Still under guarantee.
£150 ono. Call Gavin @ 088-623567, e-mail gav@alien.ie.

Wanted

1GB+ EIDE Harddisk.
Call Gavin @ 088-623567, e-mail gav@alien.ie

Using Links on the Amiga

By Barry McConnell

MOST operating systems these days include the concept of links, in one form or another. Unix, of course, has them, in the guise of the "ln" command. Macintosh System 7 has them, with its cute "Make Alias" menu item. And now the Amiga has them, using the (appropriately-named) "Makelink" Shell command.

However, the majority of people never use links on the Amiga, as a result of a few basic misconceptions:

· They just do not work. We are advised by Commodore themselves not to use them, as their implementation is flawed.

· What are they useful for? We have survived perfectly well without them up until now: how can they possibly make life easier?

The reality is, links are actually very useful. Once you know how they work and what their limitations are, working with them becomes second nature.

Different people use them for different things. For example:

· Some folk like to store MultiView somewhere other than SYS:Utilities/. They also do not like when the occasional document has its Default Tool pointing to AmigaGuide. You can create a link from both SYS:Utilities/MultiView and SYS:Utilities/AmigaGuide to (say) Work:Bin/MultiView. This takes up as much extra disk space as two zero-byte files (i.e. not much!), and whenever something goes to access either of these two files, it will be transparently redirected to the real executable. You could also, for example, make a link from SYS:C/LhA to wherever it really lives, to keep those utilities with hard-coded parameters happy.

· If you like a rarely-changing, pristine System partition, you may like to place WBStartup somewhere else. This is easy with links, as you can create a link from the SYS:WBStartup drawer to anywhere you like. This concept can be extended to any drawer that has a tendency to get large, for example my Comms:Downloads/New drawer—a repository for medium-term downloads on my system ("Cool, but I'll look at it later")—is actually a link to somewhere on Work:, on which there is far more space to play around with.

· I have a bunch of drawers left out on my Workbench that I access frequently.

However, I sometimes also find myself accessing them from within the drawers they are really stored. Links solve the problem: make a link to the drawer, and choose the Leave Out menu item on the link!

· I recently merged my SYS:System, SYS:Tools and SYS:Utilities drawers, since it is not clear what files live where. I wanted to call the new drawer SYS:Tools, but some applications expect to find certain files in SYS:System, and disk formatting is happier when the Format command lives there. One link later, and everything was happy!

There are many more uses for links, but I hope the above examples will give you a taste for them. Sure, you can survive without them, but they are another nice touch to any OS, and if this article clears up the mystery surrounding them, making your life a little easier as a result, then I will have done my job well.

Before I launch into a discussion of how links work, I think it is only fair to point out that the information presented here is based on Kickstart 3.1, as present in AGA Amigas (A1200 and A4000), and in older Amigas if you purchased the 3.1 Upgrade Kit. Much of it applies to older OS versions (from 2.0 onwards), but I do not guarantee that this is always the case. I have tried to point out the places where Kickstarts 2.0 and 3.1 differ, but I am not responsible if you lose valuable data! Always experiment with copies of files, preferably on a scratch partition.

So, at this stage you should have your Shell open, with your fingers eagerly poised. Change directory to somewhere on your hard drive, with a couple of temporary files you conven-

iently placed there beforehand. For example:

```
Work:Temp> Dir
PriMan
PriMan.info
```

Now type in the following two commands:

```
Work:Temp> Makelink NewPriMan
PriMan
Work:Temp> Dir
NewPriMan <hl>
PriMan
PriMan.info
```

You have just created a link to PriMan! Whenever something goes to open the file NewPriMan (e.g. you execute it from the Shell), the filesystem will transparently redirect it to the original PriMan file. You can tell it is a link by the <hl> after the filename. But most other things will not notice this; take, for example, the List command:

```
Work:Temp> List
Priman .     25700 —rwed
PriMan.info 2285
NewPriMan    25700 —rwed
3 files - 112 blocks used
```

Not only does it not recognise that NewPriMan is a link, but it also gets the "blocks used" entry wrong: in reality, the file only occupies minimal space on the disk.

This is why you have to be careful with links. Here we have a perfectly-working link, but something (the List command) that does not quite follow what is going on. This can lead to confusion if you are not quite sure yourself what is going on! Thankfully, most software that really needs to know about links (backup software, disk optimisers, and the like) do in fact behave correctly, and include "special cases" for links. But it is wise to check in advance: older software (and third-party filesystems) may not know about them.

The next thing you will want to try is making links to directories. Go ahead, try this:

```
Work:Temp> Makedir MyDir
Work:Temp> Makelink NewDir MyDir
```

Links to directories require use of the FORCE keyword

Oops! AmigaDOS is holding our hand here - it wants to be sure we know what we are doing. But we carry on bravely...

```
Work:Temp> Makelink NewDir MyDir
FORCE
Work:Temp> Dir
MyDir (dir)
NewDir (dir) <hl>
NewPriMan <hl>
PriMan
PriMan.info
```

Excellent! A link to a directory! But what happens if we try entering it?

```
Work:Temp> Cd NewDir
Work:Temp/MyDir>
```

Notice what happens to the Shell prompt? Like I said, links are very transparent, and what the Shell asked for, it did not quite get: we are now inside MyDir, which is what NewDir is linked to. (It could be argued that this behaviour is **not** actually transparent, but I will not pretend that the Amiga implementation of links is perfect.)

Just to really check all is running smoothly, try this:

```
Work:Temp/MyDir> Copy /NewPriMan
""
Work:Temp/MyDir> Dir
NewPriMan
```

(The double quotes mean "current directory".) Notice that the link itself is not copied (no <hl> after the directory entry!), but rather the original file. If you want to copy a link and end up with a link afterwards, you need to use the MakeLink command again. In fact, let us do just that:

```
Work:Temp/MyDir> Makelink
VeryNewPriMan /NewPriMan
```

You have to be careful that you get the argument order to Makelink correct:

while most other Shell commands (Copy and Rename, for example) create a new file named as the second argument from an existing file named as the first argument, the Makelink command creates a new file named as the **first** argument. This is logical if you think about it in a "Source, Destination" way - the link is **from** the new file, **to** the existing file.

Let us just check this is all working merrily. We will do a Dir on the directory link we created earlier, which should produce exactly the same output as we would get on a Dir in the current directory:

```
Work:Temp/MyDir> Dir /NewDir
NewPriMan
VeryNewPriMan <hl>
```

Perfect! Now, this is all working pretty much as you would expect. But what does Workbench make of it all? Try going into Work:Temp from Workbench and double-clicking on MyDir and NewDir in turn. Notice what happens? When you double-click on one, the window belonging to the other closes first! Since they are both really the same thing, Workbench refuses to let you have both open at the same time. If you choose Information on a file in NewDir, the title bar will tell you you are really looking at a file in MyDir.

This is all very well, but—back in the Shell—we are going to try something that may not behave quite as you would expect. Try this:

```
Work:Temp/MyDir> Cd /
Work:Temp> Delete PriMan
PriMan  Deleted
```

Oops! Is this a fatal mistake? No - take a look see...

```
Work:Temp> Dir
MyDir (dir)
NewDir (dir) <hl>
NewPriMan <hl>
PriMan.info
```

Our first link is still there! If you do not believe the Dir command, try executing it and see! When you make a link to a file, the "next Link" field is filled in in the original file's header, pointing to the newly-created link. If this field was already filled in, it is overwritten, after copying its contents to the "next Link" field of the new link. Thus the filesystem creates a linked list, starting at the original file, and working its way (in reverse order) through all its links.

When you go to delete this original file, the first link in the list (being the last link created) becomes the "original" file. This can be shown quite simply by:

```
Work:Temp> Dir MyDir
NewPriMan
VeryNewPriMan
```

What was once a link (VeryNewPriMan) is now a solid file. You could delete this, and NewPriMan in the parent directory would become the original; deleting this too would then leave no traces of our favourite task monitor! (Recall that NewPriMan in this directory was never a link, as it was created by the Copy command.) As you would expect, deleting a mere link just causes that link to vanish—the filesystem takes care of changing the "next Link" field of the link before it in the list to point to the link after it.

But be careful! Going on the output of the Dir command alone, you might be fooled into believing that it would be possible to completely wipe MyDir, and leave NewDir intact containing what were the contents of MyDir. But no - NewDir is only a link to MyDir, and there are not a bunch of links within NewDir to the files in MyDir. So if you delete everything in NewDir, MyDir will similarly reflect the deletions, and if you try to delete MyDir, AmigaDOS will complain, since—while you can delete a file and have its contents taken on by one of its links—this does not work for directories. (It would be pretty useless anyway, since you can only delete an empty directory.)

A word of warning: this "passing of data" from the original file to a link is badly broken under 2.0, and can cause checksum errors on your (hard) disk. There are no problems under 3.1, or when simply deleting the link.

Getting adventurous, you might try:

```
Work:Temp> Makelink
SYS:AnotherPriMan NewPriMan
object is not of required type
```

What's this then? Hardly a helpful error message! What AmigaDOS really means to say is, you are not allowed to make links across partitions. A link works by directly referencing a file (or directory) block, which only makes sense within a particular partition, as block numbers are not unique across partitions. (This is why you can even rename the original file a link points to, and not break the link—renaming the file leaves its header at the same block.) To allow cross-partition links, a new mechanism is needed: the soft link.

Soft links work similarly to hard links (i.e. the links I have just discussed), except they are a bit "looser". Once again, applications

are transparently redirected to the original file, although it is a little easier for them to notice that something is a soft link, and by the same token a little easier for them to get confused.

The difference between a hard link and a soft link is in the way they point to the original file. Recall that a hard link stores a reference to the block where you can find the original file. A soft link stores the full pathname of this file, e.g. Work:Temp/PriMan. This means the file can reside anywhere, on any partition - in fact, it can even use a logical assignment, or reside on a drive that is not even mounted, whereupon AmigaDOS will put up a "Please insert volume..." requester when you try to access it. A side-effect of this is that the link can point to something that does not even belong to the filesystem, although this is of dubious use. (Perhaps you could mimic something like /dev/audio on a Unix system?)

Soft links are where you can run into trouble quite easily, and hence the AmigaDOS Makelink command does not actually support them. You will need to get hold of either a Public Domain Makelink command, or use CShell's built-in version. Both of these are available on Aminet. Since I use CShell, the following examples are taken from it.

CShell has many built-in commands to replace AmigaDOS's, and many extra commands and functions to make it worthwhile (for me anyway) using it in preference to the original AmigaDOS Shell. However, one must be careful to note the differences between the two shells. As an example, CShell's Dir command (which can be abbreviated to ls) looks like this:

```
Work:Temp> ls
Directory of Work:Temp
MyDir
NewDir -> Work:Temp/MyDir
NewPriMan -> Work:Temp/MyDir/
VeryNewPriMan
PriMan.info
62 Blocks, 28,985 Bytes used in 4
files
```

Notice how more detailed information is given here. We actually get the full pathname alongside the two links, and a 'H' in the attributes field to show they are hard links.

CShell is also able to create soft links, using its own Makelink equivalent, which can be abbreviated to ln. This can take an optional switch '-s' to create a soft link (instead of the default hard link). To show it in action, we will use a second convenient temporary directory, to make yet another link to PriMan:

```
Work:Temp> cd System:Temp
System:Temp> ln -s PriMan
Work:Temp/NewPriMan
Work:Temp/NewPriMan: object al-
ready exists
```

Oops! This is why you need to be careful when using a different shell - CShell expects the arguments to ln in a different order to the AmigaDOS Makelink command.

```
System:Temp> ln -s
Work:Temp/NewPriMan PriMan
System:Temp> ls
Directory of System:Temp
S—rwed  0  12-Feb-95 22:16:21
PriMan -> Work:Temp/NewPriMan
```

Note the two differences here. Firstly, the 'S' indicates it is a soft link. Secondly, the file size is zero. This is because it is possible that the file does not exist yet, or is on an unmounted volume. While CShell knows its name, it does not know anything else about it, and will not try to find out until you access the file.

Links to directories work as you would expect:

```
System:Temp> ln -s Work:Temp/MyDir
NewDir
System:Temp> ls
Directory of System:Temp
S—rwed          0  12-Feb-95
22:26:34  NewDir -> Work:Temp/
MyDir
S—rwed          0  12-Feb-95
22:16:21  PriMan -> Work:Temp/
NewPriMan
4 Blocks, 0 Bytes used in 2 files
```

So far, so good. But what does Workbench make of soft links? Directories are fine - they behave exactly the same as the hard link version, i.e. you can only have one of them open at a time. But a link to a file presents Workbench with a problem: it thinks it is a directory, and complains loudly when you attempt to open it. The solution is to attach an icon to

it (by copying over a ".info" file). The same applies to the original AmigaDOS Dir command - it too thinks a soft link to a file is a directory.

While I am on the topic of icons, I'll make a comment about them now. If you are making a link to a file with an icon, it is a good idea to make a *copy* of the icon, instead of making a link to it. This is because you may decide to move the link somewhere and snapshot its icon in a different position to the original. If the icon was a link, then the original icon would be the one to get snapshotted! Since you will probably want the original and the link to be in different positions, this won't be good. So make a copy of the ".info" file!

The area where hard links and soft links differ wildly is when it comes to deletion. Since the file a hard link points to "knows" it has hard links, when you delete it, one of the links can take on its data. But a file cannot tell if it has any soft links to it! So if you delete the original and then try to access the soft link, you will get an "object not found" error, just as you would if you made a soft link to a non-existent file in the first place and then tried to access it. So be careful!

This section discusses the limitations and peculiarities of links.

You must be careful when making a hard link to a text file. If you later go to edit the file, and your text editor makes a backup copy by renaming it into another directory on the same partition, any links to that file will reference the backup copy! What your editor needs to do is copy the file (instead of renaming it), and then directly overwrite the original, to keep the links happy. Alternatively, just making a soft link will solve the problem, since soft links do not notice when a change is made to the file they are linked to.

Hard links to the root directory are not allowed (except with RAM:)—use a soft link instead. Soft links do not work at all with RAM:, by the way, and indeed under Kickstart 3.1 the (broken) code that handled soft links was removed from the RAM handler. AmigaDOS does not like making links to a directory in the path to the current directory (to avoid getting into loops), but CShell will quite happily do this. As links themselves cannot be locked, and hard links require a lock to the object to be linked, hard links to soft links are not possible either, and you will just end up with a hard link to the original file. (A hard link to a hard link is also a hard link to the original file, as discussed earlier.)

Soft links are *supposed* to be interpreted relative to the directory in which they reside, but this is not always the case, due to a bug in their design. Therefore it is best to always supply a full pathname, starting with the volume name (e.g. Work:Temp/PriMan). Staying with soft links, there is also no length checking done on the pathname to which they point, so it is actually possible to overwrite other data in the file header block! Proceed with care.

If you attach a comment to a hard or soft link, it is the original file that gets the comment (although it is still visible to the link). The same happens if you make a change to the protection bits. A file and its soft links can have different filedates, but this is not true of hard links—if you change the date on either the original or one of its links, then everything gets the new date.

Lastly, under 2.0, soft links in the middle of a pathname would not always work, but they should under 3.1.

For those who would prefer to use the Workbench to handle making links, there are several utilities that work in the same way as the Macintosh's Make Alias menu item. One of these is (funnily enough) called WBMakeLink, written by John Hughes, and available on Aminet. It adds itself to your Tools menu and makes a hard link to anything selected when you access it. It will correctly make a copy of the ".info" file, if one exists. The only problem is it does not (currently) make soft links.

That just about wraps it up. If you want more information about links, try playing around with them yourself and see what happens!

To Be or not to Be?

by Eddy Carroll

IN April 1994, Commodore International went into liquidation. Since that time, no work has been done on developing the next generation of Amigas, which is why most of us are still using the same old A4000s and A1200s (or even older machines) that we were two years ago.

Two years is a long time in the computer business—a *very* long time, in fact. In those two years, the PC market has changed quite a bit; no more is the Intel 486/66 PC considered high-end; these days, such a system is barely fast enough to cope with modern PC games, and most new users are buying entry level Pentium systems.

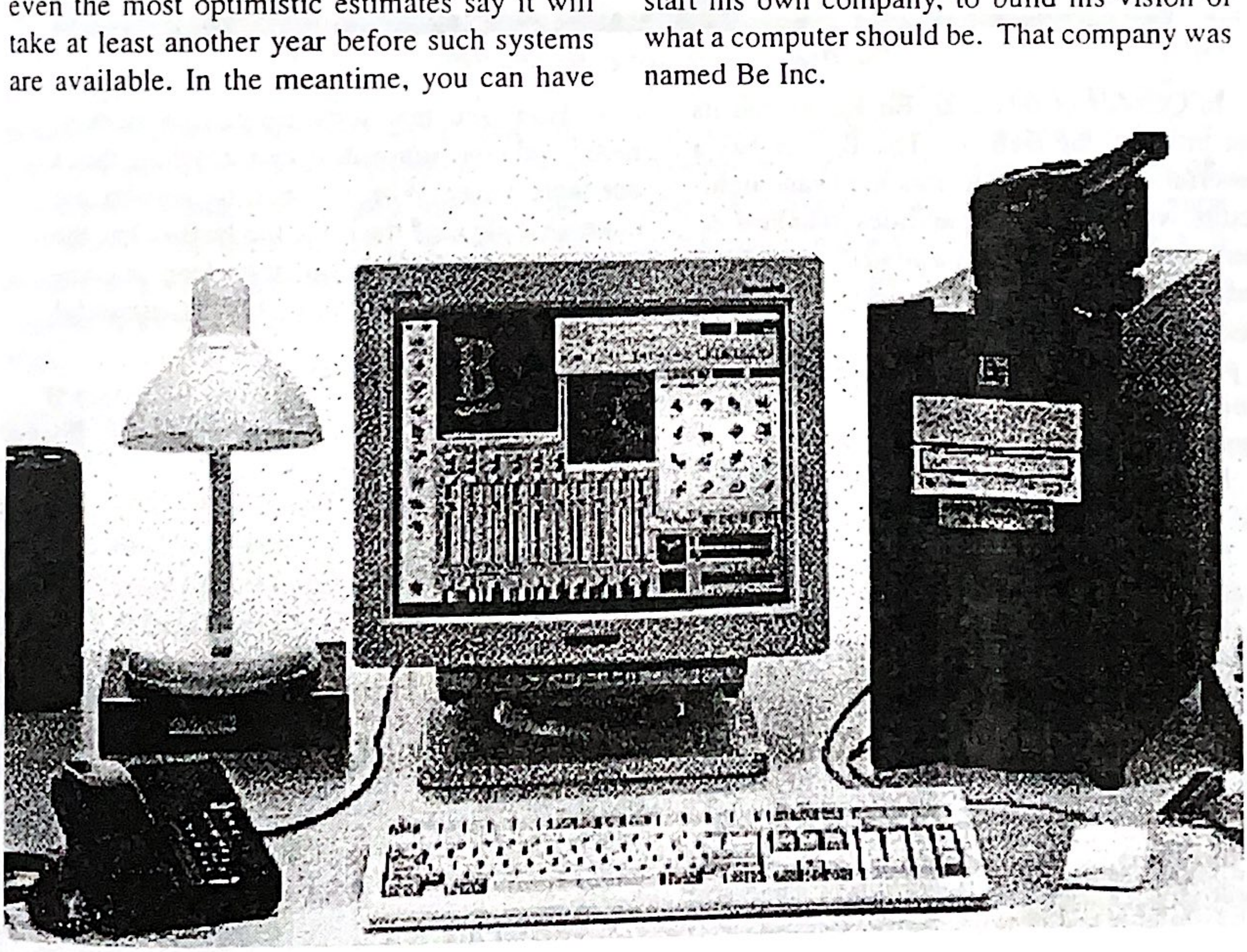
For the Amiga user who is ready to upgrade, it looks like there isn't much of a choice. Although Escom has announced their plans for the next generation of PowerPC-based Amigas, even the most optimistic estimates say it will take at least another year before such systems are available. In the meantime, you can have the stiffling "user-friendliness" of a PowerMac, or give in gracefully and jump on the Pentium bandwagon.

If, like myself, you're reluctant to give up the many benefits of the Amiga, this doesn't seem like much of a choice! What a shame there isn't an alternative...

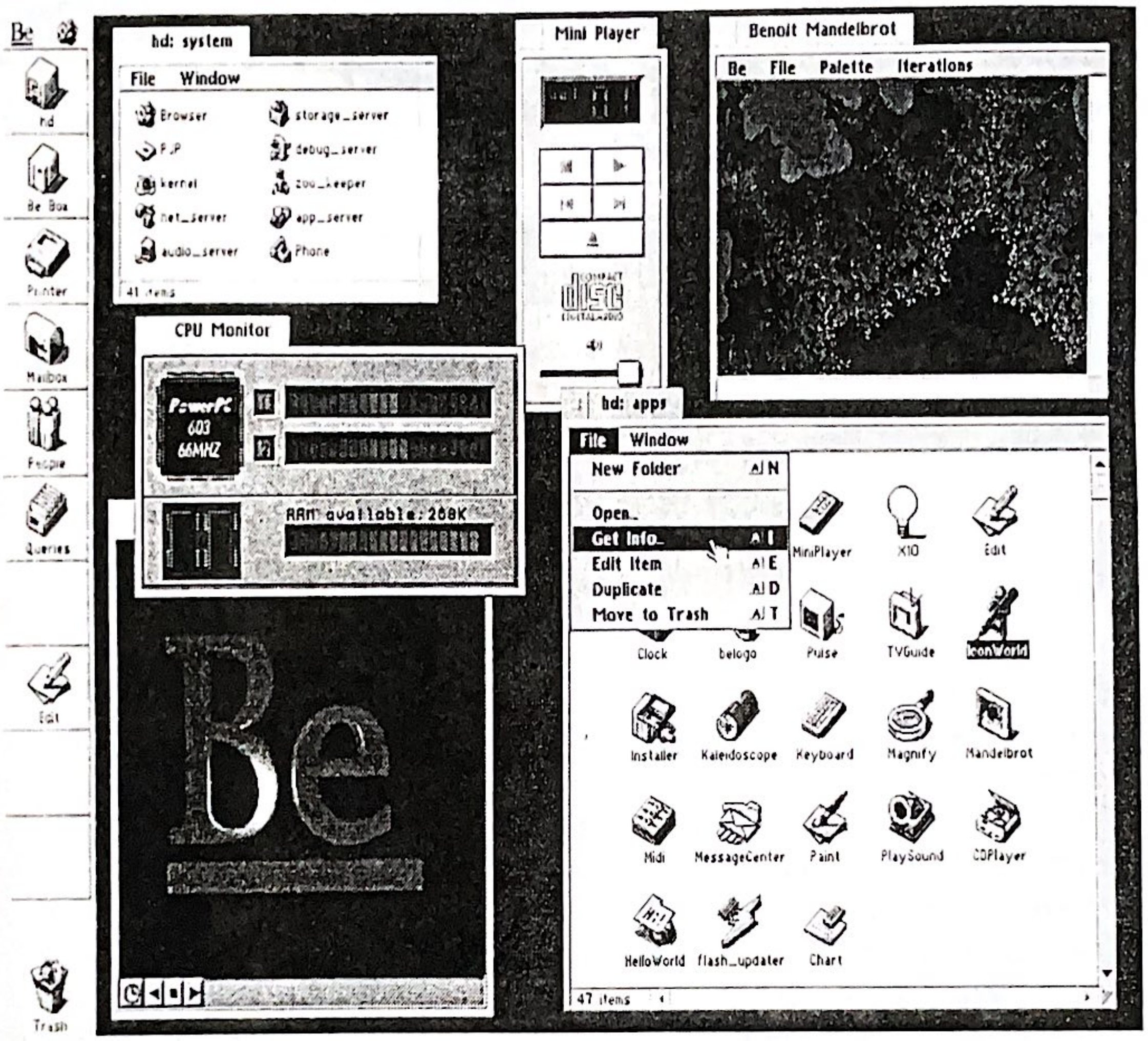
...but now there is!

A New Kid In Town

Five years ago, a fairly well known Apple personality called Jean-Louis Gassée left his prestigous job as Vice President of Product Development, at Apple, seeking a change of pace. He discussed taking over product development at Commodore with Irving Gould, but (probably sensibly, in retrospect) decided that Commodore was not for him. Instead, he decided to start his own company, to build his vision of what a computer should be. That company was named Be Inc.



Be Inc's BeBox in all its glory.



The BeBox's operating environment.

In October of this year, Be announced its first product, the BeBox. The BeBox has a powerful, flexible, multimedia hardware architecture with rich I/O capabilities standard on every unit. It also features a powerful real-time multitasking kernel, and an object-oriented class library and application framework. Which in plain english means that this is what the Amiga might have looked like, had it been designed in the 90's rather than the 80's.

In particular, the BeBox is _not_ compatible with any other system.

No DOS, no Windows, no MAC, no Unix, no extra baggage to carry around. "Well then, what's the point?" I hear you cry? The point is that this is the first new system to come out since the original Amiga which can make a clean start, unhampered by all the backwards compatibility that has constrained other systems.

Certainly, in the short term, it may take some time for a good base of software to appear. However, that software can be sleeker, faster, and more innovative than anything that's out there today. With no need to support the outmoded ideas of the 80's, the BeBox has the potential to be the next quantum leap in computing. Hyperbole? Perhaps... Fun? Definitely!

Hardware Heaven

The hardware features two PowerPC 603 processors running at 66 MHz, room for 256 MB of RAM, and 16-bit CD-quality sound. It provides a broad spectrum of expansion and I/O options, including three slots on the very fast (132 MB/sec) PCI bus for high-speed add-on cards; five expansion slots on the ISA bus for low-cost, lower-bandwidth cards; four MIDI ports; a high-speed SCSI-II port; four serial ports, one "GeekPort," two joystick ports, three infrared ports, and one parallel port. The developer system is priced at $1700 — this is a barebones unit, to which must be added memory, a keyboard, hard disc, and monitor. As such, it works

out about the same price as a similarly configured Pentium system.

The GeekPort is quite handy; it's a standard DB37 connector which contains a multitude of signals; everything from a range of voltages, to standard i/o signalling lines, to high-speed A/D and D/A converters. The intent is to make it as easy as possible for hobbyists to design add-on equipment which will easily plug into the machine.

The most innovative feature is probably the fact that the BeBox supports multiple processors: two, in the development model, and up to eight in production models. These are dyanmically allocated different tasks by the operating system, so for example, you might have one processor spending its time generating realtime fractals, while another processor looks after the operating system and windowing interface. This is multi-tasking in the truest sense of the word.

The system software is small, fast, and real-time. It provides multithreading, preemptive multitasking, and memory protection. On top of the proprietary kernel is: a graphics server that delivers ultra-responsive, continuously updated windows; a database server that supports live queries of user- or developer-defined collections of data; and a digital media toolkit that allows for manipulation of real-time streams of audio/video data. All of this is accessible from the object-oriented C++ framework.

If the above paragraph doesn't mean anything to you, then you will almost certainly not be a potential customer for the BeBox—at least, not yet. Right now, Be Inc is actively pushing the BeBox to developers only. Their intent is to generate a lot of excitement in the developer community about the machine, and to get people developing extremely cool applications that can be used to promote the machine when it goes on sale to the general public in 1996. If the buzz on Usenet is anything to go by, this strategy is already paying off big-time.

Every Silver Lining...

Of course, nothing's perfect, and the BeBox does have a few niggling little things which need fixing. In particular, the early versions of the software use fixed-size scrollbars (like Windows 3.1 and the MacOS), instead of the infinitely more sensible proportional scrollbars used on the Amiga. Also, activating a window automatically brings it to the front, possibly obscuring other windows you are interested in. The computer casing is a gaudy mix of red, white and blue, best described as "French Flag meets Andy Warhol". Nevertheless, these are all things which can be easily corrected before the official public launch next year.

But what about Windows '95, and the unstoppable juggernaut behind it (also known as Microsoft)? Surely Be Inc, as a small startup company, can't seriously expect to have any significant impact on a market controlled by a company that spends over $100M merely on a launch campaign?

Obviously, no. What Be is really targeting are the niche markets: video, audio, control, kiosks, special applications—areas that PCs are simply not very well suited for, but seem to end up being used for anyway.

It remains to be seen how Be Inc will fare in the marketplace. It has certainly got off to a promising start; however, many of you will remember the NeXT Cube from Steve Jobs, another ex-Apple person, which promised much of the same, and with little overall success. It looks likely that Be will avoid the mistakes which cost NeXT so dearly, though whether that will be enough to ensure success is open for debate.

Regardless, the BeBox looks like the consumate hacker's machine, and it's certainly the first new machine launched since the original Amiga about which I've actually got excited. Escom seem to have good plans for the Amiga's future, and perhaps they'll pull them off. I'll check back in a year or two to see if they did.

In the meantime, my developer application has already been emailed to Be Inc. And I find much comfort in the fact that Jean-Louis Gassée's personalised number plate reads "Amiga 96".

For more information, connect to website `http://www.be.com/`

The CUGI Library

by David Fitzgerald

The CUGI library has a good selection of technical books pertaining to the Amiga, including works such as the ROM Kernel Reference Manuals (for 1.3 as well as 2.0), guides to the 68000 CPU and 68881/68882 FPU, and 'C' programming references. Also of interest to programmers is the *User Interface Style Guide*, which is the official guide on creating a standard user interface.

There are also some more general books, for example, the *Desktop Video Guide* and *Amiga Tricks and Tips*. If you have one of the older Commodore machines, such as the C64 or C128, you'll find plenty of books relating to the hardware and programming of those machines.

You are probably aware that the CUGI library also has some hardware available for loan. This includes an audio sampler, a video digitiser (the VIDI-12) and a hand-held scanner. Other items of general interest are Amiga-related videos and previous issues of the CUGI newsletter.

Here is a list of what is available through the library. This list is also available on CUGI-BBS which, to remind you, recently changed number to (01) 837 0204.

Commodore Reference Manuals

Amiga ROM Kernel, Libraries (2.0)
Amiga ROM Kernel, Devices (2.0)
Amiga ROM Kernel, Includes and AutoDocs (2.0)

Amiga ROM Kernel, Style Guide

Amiga ROM Kernel (full set of Kickstart 1.3 volumes)
AmigaDOS Manual 3rd Edition
Workbench 3.0 User's Guide
Workbench 3.0 User's Guide AGA Supplement Enhancer Software (OS 1.3 reference)
Using the System Software (OS 2)
A2000, A500 Amiga BASIC Manual
A1200 User's Guide
A1200 User's Guide Service Supplement
1084S High Resolution Monitor Reference Manual
A590 Reference Manual

C Programming

Amiga C for Advanced Programmers (Abacus Books)
New C Primer Plus (The Waite Group)
The C Library (McGraw-Hill)
The C Programming Language - 2nd Edition (Kernighan and Ritchie)
Programming in C
C quick reference (Que)

General Interest

AmigaDos Inside and Out (Revised Edition also)
Amiga Disk Drives Inside and Out (Abacus Books)
Amiga for Beginners
Amiga Intern
Amiga Tricks and Tips (Abacus Books)
The Amiga System: An Introduction
Compute!'s Beginners Guide to the Amiga Amiga for Beginners (Abacus Books)
Kids and the Amiga (Compute!)
Amiga Applications (Compute!)
Amazing Computing's guide to the Amiga (summer 92)
CD-ROM (vol 2) Optical Publishing (Microsoft Press)
Looking good in print
Amiga Format Book
ICPUG Newsletters

Graphics

Becoming an Amiga Artist
Deluxe Paint II User's Manual
"Amiga World" Official AmigaVision Handbook Inside Amiga graphics (Compute!)
Using Deluxe Paint - 2nd edition (Compute!)
Amiga Desktop Video Power (Abacus Books)
Amiga Desktop Video Guide (Abacus Books)
"Video Toaster User" magazines

Programming

Amiga Guide for Advanced Programmers
Amiga World tech journal Dec 91
Amiga world tech journal Feb 92
68000 Pocketbook (Glentop)
MC68881/MC68882 Floating-Point Coprocessor User's Manual (Motorola)
Amiga Programmer's Handbook (Sybex)
Mapping the Amiga
Microcomputer Puzzles

Commodore 64/128

C64 Programmers Reference Guide
C128 Programmers Reference Guide
C64 Service Manual
C128 Service Manual
Advanced C64 Machine Code

The Anatomy of the C64
The Anatomy of the 1541

Hardware

Audio Sampler
Video Backup System
2400 baud Modem
Video Digitiser
Power Scanner

Games Software

Castle Master
Mercenary
StarFleet I
Murder!
ArmourGeddon
Gunship
TV Sports Baseball
John Madden American Football (Platinum Edition)
Indianapolis 500 (Platinum Edition)

Videos

History of the Amiga
The Deluxe Paint IV Video Guide
Amiga Animation Video Volume 2
CU Amiga's Guide to Amiga Video
Space Wars

New CD-ROM Additions

CUGI recently added two new CD-ROMs to its library. *Gold Fish* contains all 1,000 Fred Fish disks, while *Fresh Fonts* has over 100 shareware fonts, each in Adobe, Bitmap, CompuGraphic and TrueType formats.

The library also received some classic games (listed above) from Tommy Rolfs, as he left for pastures new.

Library Rules & Fees

Starting from February 1995, the charges and fines are now updated to reflect the change from bi-weekly to monthly meetings. The fine for overdue items is now £1.00 (increased from 50p). This is hardly excessive, since I've not yet had to enforce it.

Currently, charges are as follows:

Item/Category	*Deposit*	*Rental*
Sampler	£5.00	£2.00
Video Backup System	£5.00	£2.00
Modem	£5.00	£2.00
Video Digitiser	£10.00	£5.00
Scanner	£20.00	£5.00
CD-ROMs	£4.00	£1.00
Videos	£2.50	£2.50
Games	£2.00	£1.00

The deposit for the scanner is relatively high because it is a delicate piece of hardware. Since items in the library are effectively investments by the user group and members thereof, it is expected that members borrowing hardware items should be especially careful that the instructions are read and understood before use. The same guideline applies to the other items available.

Thank You

Suggestions for library services are welcome, as are donations of items for inclusion in the library! Anyone interested in any items (whether listed or not) is welcome to talk to me at any of the monthly meetings.